Biasly: a machine learning based platform for automatic racial discrimination detection in online texts

Anonymous ACL submission

Abstract

Warning: this paper contains content that may be offensive or upsetting.

Detecting hateful, toxic, and otherwise racist or sexist language in user-generated online contents has become an increasingly important task in recent years. Indeed, the anonymity, the transience, the size of messages, and the difficulty of management, facilitate the diffusion of racist or hateful messages across the Internet. The critical influence of this cyber-racism is no longer limited to social media, but also has a significant effect on our society : corporate business operation, users' health, crimes, etc. Traditional racist speech reporting channels have proven inadequate due to the enormous explosion of information, so there is an urgent need for a method to automatically and promptly detect texts with racial discrimination. We propose in this work, a machine learning-based approach to enable automatic detection of racist text content over the internet. State-of-the-art machine learning models that are able to grasp language structures are adapted in this study. Our main contribution include 1) a large scale racial discrimination data set collected from three distinct sources and annotated according to a guideline developed by specialists, 2) a set of machine learning models with various architectures for racial discrimination detection, and 3) a web-browser-based software that assist users to debias their texts when using the internet. All these resources are made publicly available.

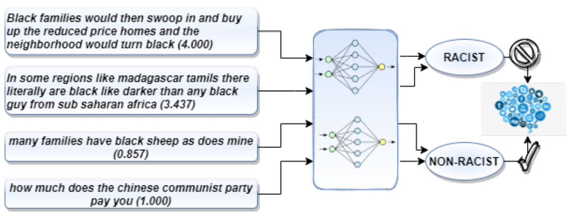

Figure 1: Racist texts are automatically detected by our system and removed from social networks

1 Introduction

Racism is a long-standing challenge in the world. It is a type of discrimination or a violent hostility towards a human group because of their skin color, their supposed race, their origin, their philosophical or religious convictions, etc (Gelber and Stone, 2007). Even if it is publicly condemned, it is often tolerated due to its virtual context on the web. Online racist speech is rapidly increasing worldwide, as nearly 60% of the world's population (estimated to be 7.8 billion by March 2020 [1]) communicates on social media. According to Alatawi et al. (2020), studies have shown that almost 53% of Americans have experienced online hate and harassment. This result is 12% higher than the results of a comparable questionnaire conducted in 2017 (Duggan, 2017). For younger teens, the results show that 21% of teens frequently encounter hate speech on social media (Clement, 2019).

As social media has become an essential part of our society today through which people communicate and exchange information on a daily basis, and through which many companies and organizations reach out to their customers to promote their products to them and ensure their satisfaction; racist texts therefore harm the experience of regular users, affect the business of online companies, and can even have serious real-life consequences (negative mental health outcomes such as depression, anxiety, and emotional stress, as well as negative physical health outcomes such as high blood pressure and low birth weight babies) (Hasanuzzaman et al., 2017). Although social media service providers now have policies to control these deviant behaviors, they are rarely followed by users (Alatawi et al., 2020). Though many providers allow users to report inappropriate contents on their platforms (Alatawi et al., 2020), many such contents may

[1] https://en.wikipedia.org/wiki/World_population

still go undiscovered due to the enormous volume of data on these platforms. Some countries have introduced restrictions on the use of social media, and others have taken legal action regarding offensive contents. However, these punishment may be not sufficient due to the anonymous nature of these platforms, which allows users to share harmful content using pseudonyms or false identities fearlessly. Assessing the levels of cyberhate on the internet would also help prevent some violence (Burnap and Williams, 2016), as perpetrators can be stopped before the violence occur by examining online messages that give strong indications of the intent to commit a crime (Alatawi et al., 2020).

Studies have focused on detecting different types of hate speech, such as detecting cyber-bullying, offensive language, antisemitism, sexism, or targeted hate speech in general (Alatawi et al., 2020). However, less attention was given to detecting racism at large scale. Even more complicated, almost all of the previous work uses small-scale and often non-public data sets. To overcome this problem of cyber-hate, among others, some organization hire specialists to analyze the contents to determine whether it is appropriate or not. In this case, undesirable content is only removed after it is published. The analysis and removal process can take days, depending on the number of analysts available and the size of the content to be analyzed. Other classical detection methods rely on blacklists and regular expressions to filter out user-posted content. These methods are inefficient since a text can contain the terms present in the said blacklists without being offensive or racist.

Given the limitations mentioned above, we propose in this work scalable models for automatic detection of racist text on the Internet. We experiment with several types of models, namely SVMs trained on the representations extracted with the bag-of-words and the TF-IDF, recurrent and convolutional models, transformer-based models such as BERT (Devlin et al., 2019) and one of its variants based on Transformers with Competitive Ensembles of Independent Mechanisms, briefly TIM (Lamb et al., 2021). These models are trained and evaluated on a newly collected dataset extracted from three data sources (Fox News, Breitbart News, Youtube) and annotated by three separate annotators according to a guideline developed by specialists. This dataset, consisting of about 80K examples, will be released for future research related to this work. We also developed a software solution that can be used by multiple users simultaneously around the world.

2 Dataset

2.1 Data collection

The dataset used in this work contains 82187 english sentences (with the average sentiment score of -0.118, ranging from -0.999 to 0.996), extracted from three sources [2] : Fox News (39408 sentences, with the average sentiment score of -0.076, ranging from -0.995 to 0.992), Breitbart News (38501 sentences, with the average sentiment score of -0.151, ranging from -0.999 to 0.991) and Youtube (4278 sentences, with the average sentiment score of -0.201, ranging from -0.99 to 0.996). Sentiment scores were calculated with the pre-trained sentiment analyzer called VADER (Valence Aware Dictionary and sEntiment Reasoner) (Hutto and Gilbert, 2014), from NLTK (Bird et al., 2009). This dataset was obtained from online news media using a programmed web crawler based on Scrapy framework [3] with all crawled data stored in PostgreSQL database in a similar way as in (Onabola et al., 2021).

2.2 Data Annotation

The sentences were annotated by freelancers using Amazon Mechanical Turk [4]. For each sentence, three annotators were assigned to give an integer value between 0 and 5, thus designating the level of racism (0 for sentences without any racial discrimination, and 5 for sentences with extreme racial discrimination). In addition, each annotator associated to the proposed score, an integer value between 0 and 10 characterizing their level of confidence for its score (0 for "not sure at all" and 10 for "extremely sure").

Thus, for each example in the dataset, we had two vectors: $[s_1, s_2, s_3]$ denoting the different scores and $[c_1, c_2, c_3]$ denoting the different confidence levels, where $s_i \in [\![0, 5]\!]$ denotes the score given by annotator number i and $c_i \in [\![0, 10]\!]$ the confidence level given by this annotator for its score s_i ($i \in \{1, 2, 3\}$). The final score was computed as

[2] This is a non-commercial research project and we may use and publish this data for research purposes only (all users of this data are required to refer to the terms of service of these data sources). The software we are developing is only a research prototype.

[3] https://github.com/scrapy/scrapy

[4] https://www.mturk.com/

the average of the scores weighted by their confidence level:

$$s = \frac{\sum_{i=1}^{3} c_i * s_i}{\sum_{i=1}^{3} c_i} \in [0, 5] \quad (1)$$

Once this score was obtained, we decided to have the final categorical label by comparing this value with a threshold value (λ) to transform the problem into a binary classification problem: the label is worth 1 if the score is greater than or equal to λ and 0 otherwise. In our work, we have chosen λ as the median value of the set of possible score, 2.5. We have also tried another approach, in which once the real score is computed as in equation 1, we round it to transform the problem into a multi-class classification problem (6 classes here, $\{0, 1, 2, 3, 4, 5\}$): if the score is in the format $a.b$, then the associated label is worth a if $b < 5$ and $a + 1$ otherwise.

2.3 Data processing

There are multiple steps in the data processing pipeline. First, we converted sentences into tokens using moses library (Hoang and Koehn, 2008). Second, we replace unicode letter and punctuation with space, remove non-printing character, lowercase, and accent. Thirdly, we proceed to tokenize and format the data using the scripts provided in Moses (Hoang and Koehn, 2008). Lastly, we used the Byte Pair Encoding (Sennrich et al., 2016) algorithm to build our vocabulary. We fixed the maximum length of sentences (after BPE) to 200, because the maximum length of sentences after pre-processing the data was 198. See table 5 for the number of BPE codes used and the vocabulary sizes. Additional information on the pre-processing steps is given in section A.3.2. For the pre-training of language models, we divided the data into three parts : 80% as training data, 10% as validation data and 10% as test data (table 4).

3 Models

3.1 Pre-training

Pre-training is the process to tune model parameters for better capturing of data latent structures usually with unlabelled data. In this study, we pre-train all our transformer based models using the masked language modelling (MLM) algorithm (Devlin et al., 2019). MLM is based on denoising auto-encoding (Vincent et al., 2008). More precisely, for a text sequence x, MLM first constructs a corrupted version $\hat{x}$ by randomly assigning to a part (e.g. 15%) of the tokens of x a special symbol $[MASK]$. The objective of the learning is to reconstruct the masked tokens $\bar{x}$ from $\hat{x}$, by minimizing the loss $-log\ p(\bar{x}|\hat{x}) \approx_{iid} -log \prod_{x_i \in \bar{x}} p(x_i|\hat{x})$ $= - \sum_{i=1}^{|x|} \mathbb{1}_{x_i \in \bar{x}}\ log\ p(x_i|\hat{x})$

The basic model here is BERT (Devlin et al., 2019). In addition to the (Vaswani et al., 2017) transformer-based BERT model, we used TIM (Lamb et al., 2021) based one, with (TIM-Comp) and without (TIM-NoComp) competition (Lamb et al., 2021). Indeed, the initial architecture of the transformer represents each position in the input information with a large monolithic hidden representation and a single set of parameters that are applied on the whole hidden representation. This does not allow for efficient learning of unrelated sources of information in the input, and limits its ability to capture independent mechanisms. To overcome this problem, TIM divides the hidden representation and parameters into several mechanisms that exchange information only through attention; and proposes a competition mechanism that encourages these mechanisms to specialize along the model training, and thus to be more independent (Lamb et al., 2021).

We evaluate our models with two metrics: MLM perplexity (ppl) and accuracy (acc). See the section A for the training setting and the number of parameters of each model. The results of the pre-training of the models are reported in the section A.4.4.

3.2 Classification

Transfer learning is a technique where a deep learning model trained on a large dataset is used to perform similar tasks on another dataset. Here, we pre-train and fine-tune our transformer-based models on the same dataset. During training, a special token $[CLS]$ is added at the beginning of each sequence. Let N be the initial sequence length (without $[CLS]$) and E the embedding dimension. The transformer-encoder will produce a latent representation $H \in \mathbb{R}^{(N+1)\times E}$, and the first element $h \in \mathbb{R}^E$ of H, corresponding to the latent representation of the classification token $[CLS]$, will be fed to the classifier.

In addition to fine-tuning our pre-trained models, we trained three other deep models: RNN (Hopfield, 1982; Graves, 2008), LSTM (Hochreiter and Schmidhuber, 1997), and CNN (LeCun et al., 1990; Lecun et al., 1998). These models directly produce a latent representations $h \in \mathbb{R}^E$ which are directly fed to the classifier. We initialized the word embedding layer of these three models with a pre-trained Glove (Pennington et al., 2014) model: glove.840B.300d [5].

A linear classification layer was added to the output of each of these models to perform classification. We used a one-layer feed forward network as classifier in our work.

For binary classification, the classifier is used to transform h to a real value p between 0 and 1 by using the sigmoid function, representing the probability of ground truth label $y \in \{0, 1\}$ being assigned to the input text x, as follows: $p = p(y = 1|x) = sigmoid(Wh + b)$ and $p(y = 0|x) = 1 - p$ where $W^\top \in \mathbb{R}^E$ and $b \in \mathbb{R}$ are parameters to optimize. The model is then trained to minimize the binary cross entropy loss $-y\ log(p) - (1 - y)\ log(1 - p)$. At test time, the resulting predicted label is compute as follows: $\hat{y} = argmax_{i \in \{0,1\}} p(y = i|x)$.

In the multiclass approach, the j-th softmax output of the neural net is $q_i = p(y = i|x) = \frac{exp(o_i)}{\sum_{j=1}^{6} exp(o_j)}$ with $o = Wh + b \in \mathbb{R}^6$ the output logit (unnormalize probability distribution) of the classifier, $W \in \mathbb{R}^{6\times E}$ and $b \in \mathbb{R}^6$ the parameters to optimize, $q \in \mathbb{R}^6$ the output softmax of the classifier (normalize probability distribution). We construct a weighted target distribution $p \in \mathbb{R}^M$ over the $M = 6$ star-values (discrete scores of the human labelers) as follows: $p_j = \sum_{i=1}^{3} \mathbb{1}_{s_i=j}\ c_i$, $p_j = \frac{p_j}{\sum_{k=1}^{M} p_k}$ to normalize. The model is trained to minimize the weighted cross-entropy loss $-C \sum_{j=1}^{M} p_j\ log\ q_j$ where $C = \frac{\sum_{i=1}^{3} c_i}{30}$. The reason for weighting by C is that if the labelers are more certain overall then we have a larger gradient (and vice-versa), and the reason for $/30$ is just for keeping the loss within a reasonable range. At test time, the resulting predicted real-valued score for how biased the input is compute as the expected score: $\hat{s} = \sum_{j=1}^{M} j\ q_j$ (to get a number between 0 and 1, we just divide by the number of stars M). In addition, we also get a confidence score about the predicted score: $\hat{c} = \frac{log(M) + \sum_{j=1}^{M} p_j\ log\ q_j}{log(M)} = 1 + \frac{\sum_{j=1}^{M} p_j\ log\ q_j}{log(M)}$ (will be 1 if the network is 100% sure about the correct score number of stars and 0 if it is completely clueless and outputs a uniform distribution).

We also trained an SVM classifier on representations extracted with BOW and TF-IDF. In this case, the ground truth label is computed as described in the section 2.2 (which corresponds to $argmax_{j \in [\![0,5]\!]} p_j$ in the multiclass approach, $p_j = \sum_{i=1}^{3} \mathbb{1}_{s_i=j}$).

[5] https://nlp.stanford.edu/projects/glove/

4 Results

4.1 Racial discrimination data collected

Figure 2 shows the distribution of classes for each data source (see also tables 13 and 12). We can notice a very low presence of extreme classes (0 and 5).

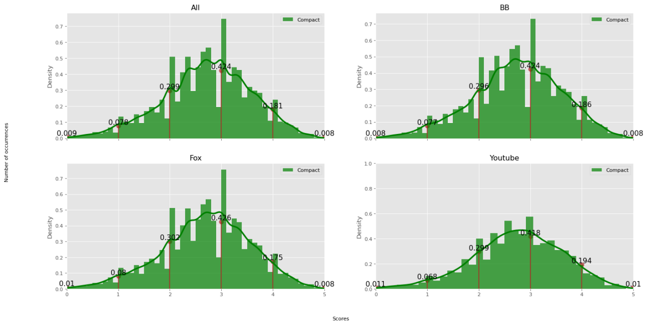

Figure 2: In green, we have the normalized histogram representing the distributions of each score. In red, the bar chart represents the percentage of occurrences of each class in the dataset.

We used Kappa coefficients (Cohen, 1960) and Krippendorff's alpha coefficient (Krippendorff, 2013) to measure the agreement between annotators. The values obtained are presented in table 3).

4.2 Model performance for bias racial discrimination classification

For simplicity, we will designate our models by the following letters : A = Bert with normal transformer pre-trained on the racially biased corpus, B = Bert with TIM-NoCom pre-trained on the racially biased corpus, C = Bert with TIM-Com pre-trained on the racially biased corpus, D = Pre-trained google Bert-base-uncased (hugging face transformers (Wolf et al., 2020)), E = RNN, F = LSTM, G = CNN, H = SVM on top of Bag of word, I = SVM on top of TF-IDF. The results obtained on test data are presented in table 1 for binary classification and table 2 for multi-class classification. These results were obtained by training the models several times with different random seeds, and taking the average of the obtained values (with the associated standard deviation).

	accuracy	F1-score
A	68.80 ± 0.04	64.69 ± 11.91
B	68.82 ± 0.03	62.56 ± 10.95
C	68.81 ± 0.04	68.93 ± 12.61
D	68.83 ± 0.00	81.53 ± 00.00
E	68.77 ± 0.04	56.20 ± 00.08
F	68.21 ± 0.03	60.50 ± 00.10
G	68.10 ± 0.01	56.71 ± 00.01
H	68.83 ± 0.00	56.12 ± 00.00
I	68.82 ± 0.00	56.12 ± 00.00

Table 1: Accuracy and F1-score for fine-tuning approach, binary classification

	accuracy	F1-score
A	43.32 ± 0.12	38.56 ± 00.16
B	43.40 ± 0.07	41.48 ± 11.70
C	43.48 ± 0.04	38.36 ± 00.08
D	43.47 ± 0.00	60.60 ± 00.00
E	43.42 ± 0.06	38.28 ± 00.11
F	43.40 ± 0.07	49.36 ± 11.24
G	43.46 ± 0.01	54.98 ± 09.72
H	43.47 ± 0.00	60.60 ± 00.00
I	43.46 ± 0.00	60.60 ± 00.00

Table 2: Accuracy and F1-score for fine-tuning approach, multi-class classification

5 Related work

Much work has been done in the past to detect hateful, offensive, and otherwise racist speech. Greevy and Smeaton (2004) use support vector machines (SVMs) (Hearst, 1998) to automatically categorize web pages as racist or not. To do this, they compared different feature representations, looking at bag-of-words (BOW) and bi-gram-of-words models, and then trained an SVM on each representation to identify the most productive method and representation for detecting racism. They obtained higher accuracy of the BOW representation on the test set than the bigrams: 87.33% versus 84.77%.

Hasanuzzaman et al. (2017) defines hate speech as "speech or expression that is likely to instill or incite hatred or prejudice against a person or group of persons based on race, nationality, ethnicity, country of origin, ethno-religious identity, religion, sexuality, gender identity or sex" (Gelber and Stone, 2007). To detect racial bias in tweets, they use demographic embeddings (Bamman et al., 2014), i.e., they focus on the demographic characteristics (age, gender, and location) of Twitter users to learn demographic word embeddings following the ideas of (Bamman et al., 2014) for geographically situated language.

Warner and Hirschberg (2012) first propose the definition of what constitutes hate speech, a definition that raises many questions to be answered in order to annotate a corpus and develop a coherent linguistic model. They use data from Yahoo! (from its newsgroup posts that readers had found offensive) and the American Jewish Congress (AJC) (consisting of pointers to websites identified as offensive). The authors begin by constructing a classifier for anti-Semitic (anti-Jewish) speech. To do this, they selected paragraphs containing words related to Judaism and Israel (9,000 paragraphs). Then, they removed some sentences: incomplete sentences, sentences with only one word or more than 64 words. Next, they identified seven (07) categories to which the labelers were to assign each paragraph: anti-Semitic, anti-black, anti-Asian, anti-woman, anti-Muslim, anti-immigrant, or other-hate. These other categories were designed to study the correlation (mutual information) between anti-Semitism and other stereotypes. The role of the labelers was therefore to assign one or more of the seven labels to each paragraph and to group South Asia, Southeast Asia, China, and the rest of Asia into the "anti-Asian" category. The anti-immigrant category was used to label xenophobic speech in Europe and the United States. The other category was used most often for anti-gay and anti-white speech, the frequency of which did not warrant its own categories. In the end, the authors had 1000 paragraphs labeled by three different annotators. The Fleiss kappa inter-rater agreement for anti-Semitic paragraphs versus other paragraphs was 0.63. They used the model-based strategy presented in (Yarowsky, 1994) to generate features from the corpus, which they later fed into an SVM classifier. In this model, each feature is dimensioned in a feature vector, with the label treated as a sign: 1 for anti-Semitic and -1 otherwise. Their best accuracy was 94%. Despite the performance obtained, this work does not focus on the issue of racism.

Nobata et al. (2016) point out that detecting language abuse is a more difficult task than one might think. Indeed, the noisy nature of the data, combined with the need for knowledge of the world, makes it not only a difficult task to automate, but also potentially difficult for people. Here are some difficulties pointed out by the authors, adapted here in the context of racist content : 1) More than just spotting keywords (intentional obfuscation of words and phrases to escape manual or automatic verification often makes detection difficult, and could lead to false positives. 2) It is difficult to track all racial or minority slurs (for example, for a blacklist-based classifier, the blacklist should not be static and should therefore be regularly updated to keep up with changing language, as some slurs that may be unacceptable to one group may be quite correct to another group, so the context of the blacklisted word is crucial (Warner and Hirschberg, 2012)); 3) Abusive language can actually be quite fluid and grammatical 5) Abusiveness or racism can cross sentence boundaries 6) Sarcasm (stinging or belligerent ironic mockery). They extracted and annotated data from three sources: Yahoo!, Finance and News. To build their classification model, they used four categories of features, namely n-grams, lexical features, syntactic features, and word-&comment-level embeddings. They found that character-level n-grams contributed the most to the accuracy of the model.

In (Burnap and Williams, 2016), the authors construct cyberhate speech classifiers for texts that target individuals or social groups based on race, gender, or handicap. Because hate crimes have been shown to increase after antecedent or "trigger" events (King and SUTTON, 2013), the authors collected Twitter data for a period immediately following the selected "trigger" events. They chose Twitter as a data source because it differs from other online social networks, such as Facebook and Google, in that the posts are widely public, programmatically accessible, and free to academic researchers. They explored a number of potential features to build their classification algorithm: bag of words, lexicon of hateful terms, and typed dependencies. Using these features, they compared SVM classification and random forest classification (with 100 trees), and found that the former performed better than the latter. They also compared using classifiers trained on each category of hate speech to using a single classifier trained on data covering all categories. As expected, the specialized classifiers performed better than their multi-category counterparts.

The authors of (Alatawi et al., 2020) work on the detection of white supremacist tweets. To do so, they collected about 1M tweets from white supremacist accounts and tagged about 2000 subsets of the data corpus to build a white supremacist dataset. Their first proposed approach uses embedding domain-specific words learned from the corpus and then classifies the tweets using a two-way LSTM: this approach yielded F1 scores ranging from 49.2% to 74.8% depending on the corpus. Their second approach uses a pre-trained linguistic model that is fine-tuned on the white supremacy dataset using the dense layer of the neural network: the F1 scores of the BERT linguistic model range from 58.7% to 79.6%.

The major limitations of these works are as follows. First, they do not make their data and their solution public (source codes, pre-trained models, etc). Second, the data in question can be considered obsolete to answer the current problematic, since the forms of racism (or more globally of discriminations) and their manifestation evolve with time. Third, the authors do not report any deployment of their solution for public use.

Many other works have focused on the development of new datasets: SOCIAL BIAS FRAMES, Reasoning about Social and Power Implications of Language (Sap et al., 2020), REALTOXICITYPROMPTS: Evaluating Neural Toxic Degeneration in Language Models (Gehman et al., 2020), BiasCorp (Onabola et al., 2021), etc. Our work directly follows the one of (Onabola et al., 2021), since our dataset is annotated along the same guideline as their.

6 Discussion

In this work, we proposed methods automatic detection of racial discrimination in online text content. To do this, we have studied several models, namely SVMs trained on the representations extracted with the bag-of-words and the TI-IFD, recurrent (RNN and LSTM) and convolutional (CNN) models, transform-based models such as BERT (Devlin et al., 2019) and one of its variants based on TIM : TIM-Com and TIM-NoCom (Lamb et al., 2021). These models are trained and evaluated on datasets automatically extracted from three data sources (Fox News, Breitbart News, Youtube) and annotated according to a guideline developed by specialists. Technical details about the software solution developed are given in the appendix.

References

Hind Saleh Alatawi, Areej Maatog Alhothali, and Kawthar Mustafa Moria. 2020. Detecting white supremacist hate speech using domain specific word embedding with deep learning and bert.

David Bamman, Chris Dyer, and Noah A. Smith. 2014. Distributed representations of geographically situated language. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 828–834, Baltimore, Maryland. Association for Computational Linguistics.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

Pete Burnap and Matthew Williams. 2016. Us and them: identifying cyber hate on twitter across multiple protected characteristics. *EPJ Data Science*, 5.

J. Clement. 2019. Percentage of teenagers in the united states who have encountered hate speech on social media platforms as of april 2018, by type.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46.

Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. 2012. L2 regularization for learning kernels.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

M. Duggan. 2017. Online harassment 2017. pew research center.

Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A. Smith. 2020. Realtoxicityprompts: Evaluating neural toxic degeneration in language models.

K. Gelber and A.S.A. Stone. 2007. *Hate Speech and Freedom of Speech in Australia*. Number vol. 2118 in Hate Speech and Freedom of Speech in Australia. Federation Press.

Alex Graves. 2008. Supervised sequence labelling with recurrent neural networks. In *Studies in Computational Intelligence*.

Edel Greevy and A. Smeaton. 2004. Classifying racist texts using a support vector machine. In *SIGIR '04*.

Mohammed Hasanuzzaman, Gaël Dias, and Andy Way. 2017. Demographic word embeddings for racism detection on Twitter. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 926–936, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Marti A. Hearst. 1998. Support vector machines. *IEEE Intelligent Systems*, 13(4):18–28.

Dan Hendrycks and Kevin Gimpel. 2020. Gaussian error linear units (gelus).

Hieu Hoang and Philipp Koehn. 2008. Design of the Moses decoder for statistical machine translation. In *Software Engineering, Testing, and Quality Assurance for Natural Language Processing*, pages 58–65, Columbus, Ohio. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9:1735–80.

J J Hopfield. 1982. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558.

Clayton J. Hutto and Eric Gilbert. 2014. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *ICWSM*.

Ryan King and GRETCHEN SUTTON. 2013. High times for hate crimes: Explaining the temporal clustering of hate-motivated offending. *Criminology*, 51.

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A method for stochastic optimization.

K. Krippendorff. 2013. describes the mathematics of alpha and its use in content analysis since 1969. page 221–250.

Alex Lamb, Di He, Anirudh Goyal, Guolin Ke, Chien-Feng Liao, Mirco Ravanelli, and Yoshua Bengio. 2021. Transformers with competitive ensembles of independent mechanisms.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Yann LeCun, Bernhard Boser, John Denker, Donnie Henderson, R. Howard, Wayne Hubbard, and Lawrence Jackel. 1990. Handwritten digit recognition with a back-propagation network. In *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2018. Focal loss for dense object detection.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

Chikashi Nobata, Joel R. Tetreault, A. Thomas, Yashar Mehdad, and Yi Chang. 2016. Abusive language detection in online user content. *Proceedings of the 25th International Conference on World Wide Web*.

Olawale Onabola, Zhuang Ma, Yang Xie, Benjamin Akera, Abdulrahman Ibraheem, Jia Xue, Dianbo Liu, and Yoshua Bengio. 2021. Hbert + biascorp – fighting racism on the web.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. pages 311–318.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter.

Maarten Sap, Saadia Gabriel, Lianhui Qin, Dan Jurafsky, Noah A. Smith, and Yejin Choi. 2020. Social bias frames: Reasoning about social and power implications of language.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. arXiv:1706.03762.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, page 1096–1103, New York, NY, USA. Association for Computing Machinery.

William Warner and Julia Hirschberg. 2012. Detecting hate speech on the world wide web. In *Proceedings of the Second Workshop on Language in Social Media*, pages 19–26, Montréal, Canada. Association for Computational Linguistics.

Jason Wei and Kai Zou. 2019. Eda: Easy data augmentation techniques for boosting performance on text classification tasks.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2020. Xlnet: Generalized autoregressive pretraining for language understanding.

David Yarowsky. 1994. Decision lists for lexical ambiguity resolution: Application to accent restoration in spanish and french. In *Proceedings of the 32nd Annual Meeting on Association for Computational Linguistics*, ACL '94, page 88–95, USA. Association for Computational Linguistics.

A Appendix

A.1 Limitations and Risks

Limitations The efforts of this work were focused more on the development of the dataset than on the development of new model architectures for detecting racial discrimination in texts, and the results obtained demonstrate the need for future work on specialized models for this task on this dataset. Indeed, of all the models used in this work, none significantly outperformed the others in terms of accuracy.

Risks The performance of the models were not perfect, therefore, applying in real world use cases will lead to certain level of inaccuracy.

A.2 Chrome extension

The detection system is made of two components: a browser extension itself and a Web API on which our model is deployed. The extension retrieves the text and sends it to the API via an http request. The API queries the model and sends back the answer that the extension displays.

A.3 Dataset

A.3.1 Agreement between our annotators

- A1 : we consider the scores without the degrees of confidence

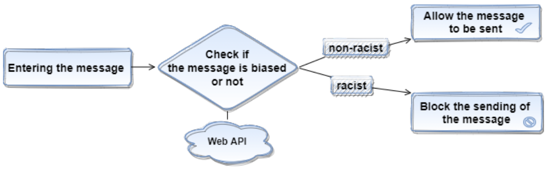

Figure 3: Activity diagram representing the functioning of our system

- A2 : we consider the annotators two by two and compute their final score with the formula of the equation 1 (average of the scores weighted by the degrees of confidence)

- A3 : we proceed as described in the first dash, but considering three possible classes for each annotator: if the confidence level is strictly lower than 5, return 2 (unknow), otherwise return 1 if the score is higher than 2.5 (biased sentence) and 0 otherwise (unbiased sentence)

	(1, 2)	**(1, 3)**	**(2, 3)**
A1	0.03, 0.05	0.04, 0.05	0.03, 0.05
A2	0.21, 0.46	0.21, 0.47	0.21, 0.47
A3	0.0, 0.07	0.06, 0.08	0.07, 0.09

Table 3: Agreement between our annotators : for any pair (a, b), a represents the Kappa coefficient and b the Krippendorff's alpha coefficient

A.3.2 Cleaning details

The following pre-processing steps were applied to the data as they are neither necessary for model pre-training nor for racial bias classification :

- the sentences have been put in lower case letters

- unicode characters (like <u+00a0> <u+00af>, <u+00a0> <u+00b0> ...) and html taged have been removed

- signs of punctuation have been replaced by the space

- url, email, phone number, number, digits and currency symbol have been replaced respectively by the symbols <url>, <email>, <phone>, <number>, <digit>, <cur> (thanks to the clean-text [6] library)

[6] https://github.com/jfilter/clean-text

We also removed nearly 3265 sentences of length < 2 for the data used to pre-train the models for the MLM task.

The maximum sentence length over the whole corpus has decreased from 369 to 198 after these steps.

A.4 Pre-training of language models

A.4.1 Data and vocabulary size

	Train (80%)	**Test (10%)**	**Valid (10%)**
All	34985	4374	4373
Fox	16784	2098	2098
BB	16415	2052	2052

Table 4: Statistics on training, validation and test data; for pre-training, according to the sources considered

Datasouce	**#BPE code**	**vocab. size**
All	2000 ($\sim$ small)	2054
All	20000 ($\sim$ large)	17984
Fox	500	572
BB	500	571
Youtube	100	117

Table 5: Size of the BPE vocabulary as a function of the number of PBE codes

A.4.2 Training setting

- Stopping criterion: terminate the experiment if the stopping criterion (we used the MLM perplexity) on the validation data does not improve for 10 consecutive epochs

- Validation metrics (when to save the best model) : each time the perplexity on the validation data improves, we saved the model parameters as the best version of the model

- fraction of words for which we have to make a prediction : 0.15

- fraction of words to mask, keep and randomize among the words to predict : 0.8, 0.1 and 0.1 respectively

- optimizer : we used Adam (Kingma and Ba, 2017) with initial learning rate of 0.0001

- maximum norm for gradient clipping (Pascanu et al., 2013) : 5

- activation : Gelu (Hendrycks and Gimpel, 2020)

- dropout rate (Srivastava et al., 2014) : 0.1
- Bert (we trained the Bert-base model): 12 layers, 12 attention heads and an embedding/hidden dimension of 768 (H)
- hidden dimension of feed forward layers : 2048
- TIM: following (Lamb et al., 2021), we replaced all layers of the transformer encoder, except the first two layers and the last layer, by TIM layers ($n_s = 2$ mechanisms, $H_c = \frac{n_{heads}}{n_s}$)

A.4.3 Number of Parameters

NT : Normal Transformer (Devlin et al., 2019)

a. **Models trained on the entire data set**

Models	2000 codes	20000 codes
NT	68142086	80392256
TIM-NoComp	**38054168**	**50304338**
TIM-Comp	48712472	60962642

Table 6: Number of parameters per model as a function of the number of BPE codes used on the whole dataset

b. **Models trained on separate sources**

See the table 7. We did not focus on Youtube in the pre-training because its size was too small for this task.

Models	Fox	BB
NT	67002428	67001659
TIM-NoComp	**36914510**	**36913741**
TIM-Comp	47572814	47572045

Table 7: Number of parameters per model for each data source

A.4.4 Results

These are the acronyms used in the following tables : ST for "Stopped Converging" (after that the model stopped converging), NBVS for "Not a better validation score" (when the model has not improved over a number of epochs), $\downarrow$ for "smaller is better" and $\uparrow$ for "higher is better". The results are presented in tables 8, 9, 10 and 11.

A.5 Classification

A.5.1 Data : class distributions

See tables 13 (multi-class classification) and 12 (binary classification).

A.5.2 Models

We trained two-layer bidirectional recurrent models, with a hidden dimension of 256 for the RNN (also CNN) and 75 for the LSTM (with 256 the overfitting was too high). Concerning the convolutional models, we used 100 output channels and kernels of sizes 3, 4 and 5 (3, 4 and 5-gram).

We also used BOW and TF-IDF to build vector representations of our data (vectorization), then trained a Linear Kernel SVM classifier on these features. For BOW, we set the vocabulary size to 20000. We used scipy.sparse [7] to store the extracted representations with BOW because of their sparse nature. For TF-IDF, we use class TfidfVectorizer [8] from scikit-learn, and the training corpus to train a vectorizer. We filtered out too rare words (occur less than in 5 titles) and too frequent words (occur more than in 90% of the sentences). We use bigrams along with unigrams in our vocabulary.

A.5.3 Experiment settings

We evaluated our models using accuracy, f1-score We use Adam (Kingma and Ba, 2017) as optimizer, with an initial learning rate of 0.0001 and weight decay (Cortes et al., 2012) rate of 0.01 for all our models. The loss function here is the (binary, for the binary version of our classification task) cross-entropy loss with logits. We used a dropout (Srivastava et al., 2014) probability of 0.1 for all our deep models, and Pytorch (Paszke et al., 2017) as a framework. The models were trained on one 48GB NVIDIA Quadro RTX 8000 GPU and a 11GB NVIDIA RTX 2080 Ti.

A.5.4 Additionnal Experiments

In addition to fine-tuning, we tried feature extraction, in the in-distribution testing setting. In this approach, no model among the pre-trained models really outperformed the other: indeed, with this approach, all models underfitted the data and failed in the generalization of the task.

In the fine-tuning approach (direct training for non-pre-trained models), despite adjusting the values of the hyperparameters (dropout, number of parameters), the following models also underfitted the data: RNN, CNN, BERT, and TIM-Com. On the other hand, LSTM and TIM-NoCom have overfitted the data.

[7] https://docs.scipy.org/doc/scipy/reference/sparse.html

[8] sklearn.feature_extraction.text.TfidfVectorizer.html

Models	Epochs	time (hh:min:ss)	MLM-accuracy ↑		MLM-perplexity (likelihood) ↓		Remark
			Test	Valid	Test	Valid	
NT	8	03:24:29	29.773	30.030	39.792	39.438	ST
TIM-NoComp	8	**02:42:15**	32.238	32.706	32.432	32.417	-
	11	03:43:16	35.719	**36.268**	28.240	27.955	ST
TIM-Comp	8	03:41:28	30.665	31.340	35.592	34.886	-
	11	05:04:44	**36.270**	**36.163**	**26.937**	**27.281**	ST
	-	06:55:19	**38.032**	**37.760**	32.395	32.058	NBVS (8/10)

Table 8: MLM accuracies and perplexities for the models trained on all data, with 2000 BPE codes

Models	Epochs	time (hh:min:ss)	MLM-accuracy ↑		MLM-perplexity (likelihood) ↓		Remark
			Test	Valid	Test	Valid	
NT	6	01:23:14	18.145	17.699	315.074	320.499	ST
TIM-NoComp	6	**01:09:33**	**20.041**	**19.896**	**218.319**	**221.323**	ST
TIM-Comp	6	01:36:14	19.591	19.625	230.818	230.643	-
	8	02:08:22	**21.444**	**21.500**	**213.367**	**219.918**	ST

Table 9: MLM accuracies and perplexities for the models trained on all data, with 20 000 BPE codes

Using these two models that overfitted the data (LSTM and TIM-NoCom), the cross-validation technique (k-fold, with k equal to 20% of the data) and a higher dropout probability (around 0.5), we observed a reduction of the overfitting, but not a real increase of the models' performance. We observed the same phenomenon when reducing the size of the model (a smaller size of the hidden representations precisely, and fewer layers of transformer encoders and multi-headed attention in the particular case of TIM-NoComp).

We also used **EDA**, Easy Data Augmentation Techniques (Wei and Zou, 2019), to artificially augment our data: deletions, replacement, and permutation of some words, as well as replacement of words by their synonyms. This technique degraded the performance of our models.

Since the dataset was unbalanced in terms of classes, we practiced up-sampling (down-sampling was impossible here, because doing so, we end up with a dataset of 6x687 = 4122 examples, since the class with the smallest number of occurrences appears 687 times in the dataset, class 5). To this, we coupled the use of FocalLoss (Lin et al., 2018) (a variant of cross-entropy that takes into account the distribution of classes) and weight training (penalizing majority classes while overestimating minority ones). This had the effect of adjusting the confusion matrix without contributing to the increase in performance.

Among the above-mentioned techniques, many others have been used to try to combat these phenomena preventing the proper training of our models, but have not changed much on the final performance. For example, we have, among other things:

- fine-tuning only k layers in the case of transformer-based models: we varied k, sometimes selecting the last k layers, sometimes selecting k layers randomly.
- combine classification and pre-training: one approach where we do a pre-training step, then a classification step and so on; another where we combine both simultaneously, i.e. the sentence with the masked tokens is passed to the transformer encoder and then the obtained representation is sent to two different classification layers, one for racial bias classification and the other for MLM.
- add one-dimensional bottleneck layer before the classification layer, to force the different classes to share information between them.
- add noise in the data: replace some tokens randomly by others during training, noisy some outputs.

Models	Epochs	time (hh:min:ss)	MLM-accuracy ↑		MLM-perplexity (likelihood) ↓		Remark
			Test	Valid	Test	Valid	
NT	13	02:35:21	16.846	17.498	81.249	78.244	ST
TIM-NoComp	16	04:24:37	38.459	38.441	16.144	17.329	-
	20	05:31:46	42.913	43.168	13.710	14.437	ST
TIM-Comp	16	**03:10:50**	**40.307**	**39.912**	**14.794**	**15.571223**	-
	22	**04:22:13**	**46.617**	**46.559**	**11.734**	**12.142027**	ST

Table 10: MLM accuracies and perplexities for the models trained on Fox

Models	Epochs	time (hh:min:ss)	MLM-accuracy ↑		MLM-perplexity (likelihood) ↓		Remark
			Test	Valid	Test	Valid	
NT	13	02:35:21	16.846	17.498	81.249	78.244	ST
TIM-NoComp	13	**02:05:34**	**37.336**	**37.936**	**17.637**	**17.557**	-
	21	03:22:46	44.102	45.215	14.302	13.887	ST
TIM-Comp	13	02:45:40	32.437	32.498	23.226	23.161	-
	21	04:28:23	40.856	41.598	16.401	16.006	-
	-	-	-	-	-	-	ST

Table 11: MLM accuracies and perplexities for the models trained on BB

Sources \Classes	0	1	Total
Fox	14842	24566	39408
BB	14130	24371	38501
Youtube	1545	2733	4278
Total	30517	51670	82187

Table 12: Class distributions for binary classification

0	1	2	3	4	5	Total
375	3139	11898	16781	6890	325	39408
308	2982	11404	16314	7173	320	38501
46	292	1278	1790	830	42	4278
729	6413	24580	34885	14893	687	82187

Table 13: Class distributions for multi-class classification

- study the dataset in order to remove the sentences that seemed similar according to the BLEU score (Papineni et al., 2002)
- remove all processing steps done on the data.
- try BERT variants (Roberta (Liu et al., 2019), XLNet (Yang et al., 2020), distilBert (Sanh et al., 2020), Albert (Lan et al., 2020)).

All these techniques did not help to improve the performance. On the other hand :

- by removing the extreme classes (0 and 5) in the data, class which were not enough represented, we obtained the performances of the first row of table 14 (4 classes classification).
- by replacing the extreme classes by their closest classes (0 by 1 and 5 by 4), then by replacing each class i by $i-1$, we obtained the performances of the second row of table 14 (classification with 4 classes also).

	accuracy	F1-score
F	56.643	72.321
F	55.6825	71.533

Table 14: Accuracy and f1-score for LSTM in 4 classes classifications setting

A.6 Some examples from the data set

We present in the table 15 some examples from the dataset, for different sources with different levels of racial bias.

In this table, s is calculated as shown in equation 1, i.e.

$$s = \frac{c_1 \times s_1 + c_2 \times s_2 + c_3 \times s_3}{c_1 + c_2 + c_3}$$

Examples	s	(s_1, c_1)	(s_2, c_2)	(s_3, c_3)	**src**
What are they going to tell Hispanic voters in Florida about rejecting a Latina for the high court? Extend that thought out to the whole country.	4.43	(4, 3)	(5, 10)	(4, 10)	Fox
Wray noted that in 2014 Chinese hackers stole more than 21 million records from the U.S.	0.30	(3, 2)	(0, 8)	(0, 10)	Fox
Yet another black thug, struggling with police, going for a weapon, and shot in self defense.	3.61	(3, 9)	(4, 6)	(4, 8)	Fox
The white liberal is the worst enemy to America, and the worst enemy to the black man.	0.00	(0, 1)	(0, 10)	(0, 10)	Fox
All chinese should be advised to leave the usa.	5.00	(5, 10)	(5, 5)	(5, 10)	Fox
And my wife who was a Jew from the Island went to Buff State and they called that Tel Aviv Tech....	4.77	(5, 9)	(4, 4)	(5, 5)	Fox
I was able to stay sequestered at home during the pandemic was because black women stocked the grocery shelves.	4.75	(4, 6)	(5, 10)	(5, 8)	BB
ASK THE CHINESE HOTEL WORKERS ABOUT MOOCH'S MOTHER ORDERING THEM AROUND LIKE YELLOW SLAVES	4.70	(5, 10)	(5, 9)	(4, 8)	BB
Several Orthodox Jewish men who came to the protest to show solidarity with the Black Lives Matter movement was assaulted and taunted with antisemitic slurs.	4.63	(3, 4)	(5, 10)	(5, 8)	BB
Why doesn't she use the money she's made off of whites to help black communities?	0.00	(0, 10)	(0, 10)	(0, 10)	BB
I am Catholic and stand shoulder to shoulder with the Jewish people and the State of Israel.	0.92	(0, 10)	(3, 8)	(0, 8)	BB
Black soldiers in the Confederate militias were paid the same as their White counterparts, were provided with provisions and carried the banners of their regime.	2.61	(4, 4)	(4, 3)	(1, 6)	BB
@McJohnson First of all, if you think those riots are "protests" then you should probably take off those ridiculous sunglasses and see what's really going down.	2.27	(3, 5)	(2, 8)	(2, 5)	Yt
Maaan I've seen looting vids but never got to see this close and the people looting really have this stupid ass look on their face literally they give off a stupid fuck auro that I can feel thru the phone. Idiots. Screaming BLM and break into black own shops.	0.00	(0, 1)	(0, 10)	(0, 10)	Yt
V.D. Hanson is delusional, he is speaking how great America was in its beginnings, how great of an idea it was etc. He continues to propagate all those myths that envisage American exceptionalism, which led to wars and conflicts and cost lives of many Americans. He ignores the Indian massacres Bear River, San Creek, etc. or Mexican wars and taking a huge part of their territory when USA was forming. He does not mention how aggressive US foreign policy was in XX century, Iran, Iraq, S. America, as if those things had never happened. Of course he is fully aware of them given his education.	0.2	(0, 10)	(1, 5)	(0, 10)	Yt
I'm enjoying the NBA. Thank you, Black athletes, for standing up against racism. We know that you hate Black athletes so stop with the BS. You should tell that to the other side.	5.0	(5, 7)	(5, 10)	(5, 7)	Yt

Table 15: Some examples from the dataset (src = sources, BB = Breitbart, Yt = Youtube)